Учебный центр при МГТУ им. Н.Э. Баумана

Специалист.ru

Системный анализ

Направление Курсы Бизнес-приложения 16 ак.час.

О преподавателе

Белова Юлия Григорьевна

- Преподаватель курсов по Системному анализу
- Бизнес аналитик в Spotware Systems
- Сертифицированный аналитик требований (IREB Certified Professional for Requirements Engineering)
- Сертифицированный бизнес-аналитик (IIBA Certification of Capability in Business Analysis)

Описание курса

«Системный анализ»

16
ак. ч.

- Как готовить документацию для разработки АС?
- Как проектировать интеграционные потоки данных?
- Как исследовать АС?
- Как работать со стейкхолдерами проекта?

Полученные знания востребованы для профессий бизнес-аналитика и системного аналитика

Целевая аудитория

- Junior аналитики (бизнес и системные)
- Люди, желающие войти в профессию системного аналитика

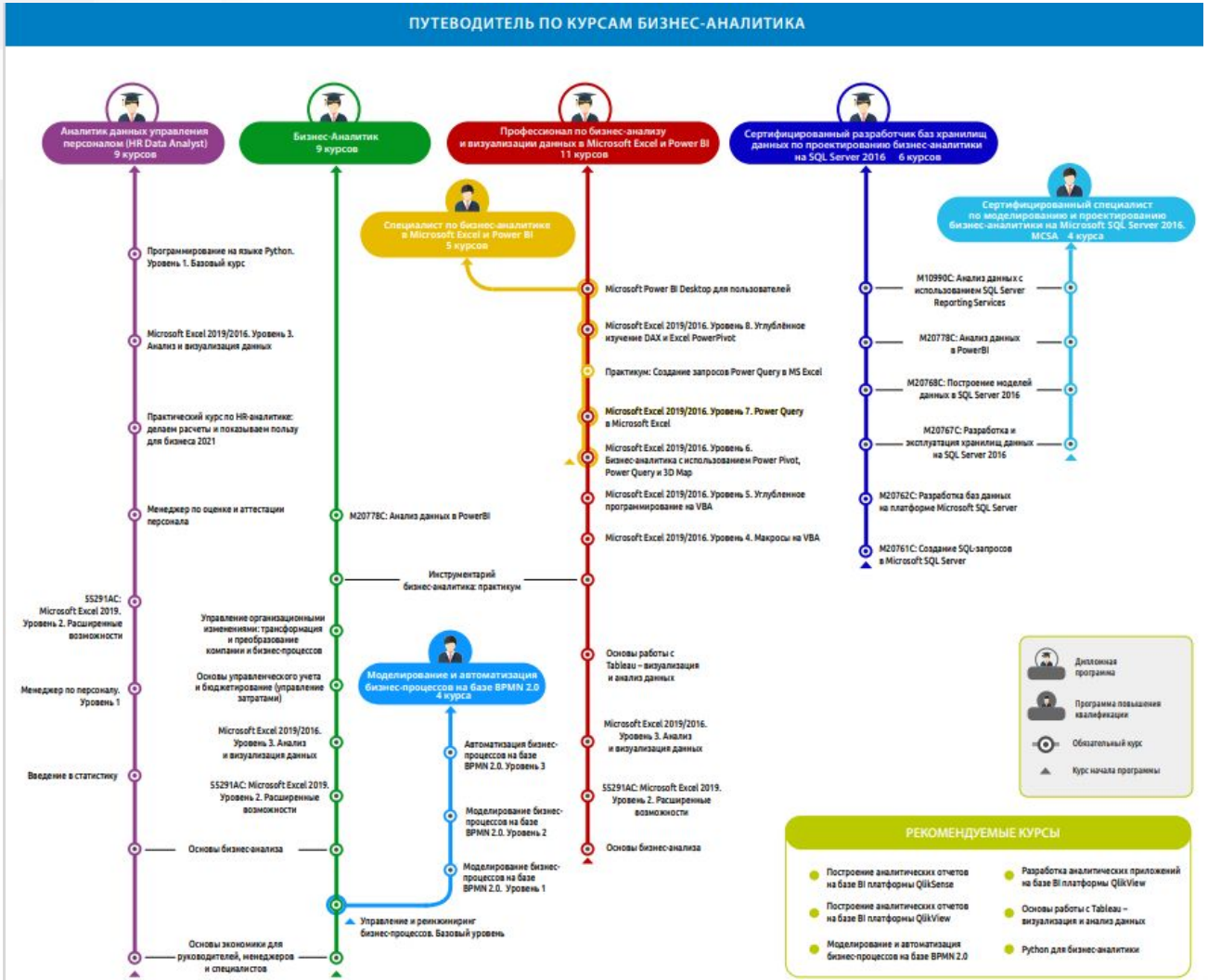
Что дает курс

- Понимание, в каком направлении развиваться в профессии
- Теоретическая и практическая база для прохождения собеседования
- Практические навыки в виде решения конкретных задач
- Интеграция полученных знаний в виде задания по подготовке ТЗ
- Готовое проверенное ТЗ для портфолио

Предварительная подготовка

- Для успешного прохождения курса по Системному анализу слушатели должны обладать следующими знаниями и навыками:
  - Успешное окончание курса «Основы программирования и баз данных».
  - Успешное прохождение курса по одной из реляционных СУБД (MySQL/PostgeSQL/MS SQL/DB2 и др)

Место в цепочке курсов



Организационные вопросы

- Оптимальный вариант курса 2х8 ак.час
- Перерывы
  - 11:30-11:40
  - 13:10:14:00
  - 15:30-15:40
- Формат вебинара
- Три практических задания — подготовка задачи, описания API, ТЗ
- Материалы, выдаваемые слушателям: методическое пособие (электронное).

Для работы нужно скачать (https://notepad-plus-plus.org/downloads/) Notepad ++ и устновить к нему плагины XML Tools, JSON Viewer

Модуль 1

Профессия системный аналитик

1
ак. ч.

- Отличие от бизнес аналитика и BI аналитика
- Виды системный аналитиков
- Основные обязанности и навыки
- Компетенции
- Понятие системы

Модуль 1

Отличие от бизнес аналитика и BI аналитика

- BI аналитик — акцент на данных, статистических параметрах и визуализации с целью подготовки отчетов для обеспечения принятия бизнес решений
- Бизнес аналитик — акцент на бизнес процессах, их описании и оптимизации, а также на описании бизнес требований.
- Системный аналитик — акцент на конкретных ИТ решениях (с учетом надежности, безопасности, производительности) и точном описании алгоритмов, подготовка технических заданий (ТЗ) для разработчиков. Основная задача — проектирование подходов к автоматизации бизнес-процессов на основе выявленных требований. Альтернативное название — Requirements Engineer — инженер требований.
  - Роли системного и бизнес аналитика часто совмещаются

Модуль 1

Виды системный аналитиков

- Специалист по конкретному типу ИТ систем /консультант (Siebel, Murex, SAP, 1С)
- Специалист по технологиям (мультимедиа, IoT, блокчейн)
- Специалист по отраслям (финансы, ритейл, игры)

Модуль 1

Основные обязанности системного аналитика

- Управление требованиями (сбор, декомпозиция, приоритезация, согласование, документирование)
- Подготовка ТЗ и прочей документации по проекту
- Ведение задач и контроль за их реализацией (совмещение роли менеджера)
- Тестирование (совмещение роли тестировщика)
- Обучение пользователей
- Работа с инцидентами (совмещение роли техподдержки)
- Консультирование и анализ по возможностям изменения системы
- Полный список можно посмотреть в Стандарте профессии

Модуль 1

Компетенции системного аналитика

- Знание технологий и конкретных решений:
  - Основы операционных систем(Linux, Windows) и сетей
  - Протоколы и стандарты веб технологий: HTTP, XML, JSON
  - Реляционные БД и SQL
  - Основы объектно-ориентированного программирования (Java/C#/Python и др.)
- Коммуникации(устные и письменные) и управленческие навыки (приоритезация и контроль исполнения задач, решение конфликтов, проактивность)
- Инструментарий моделирования (UML, BPMN)
- Исследование и анализ информационных систем
- Владение инструментами для упрощения работы (Jira, Confluence,MS Visio, GIT, Notepad++)
- Знание специфики ИТ системы с которой работаете (приходит только с опытом)

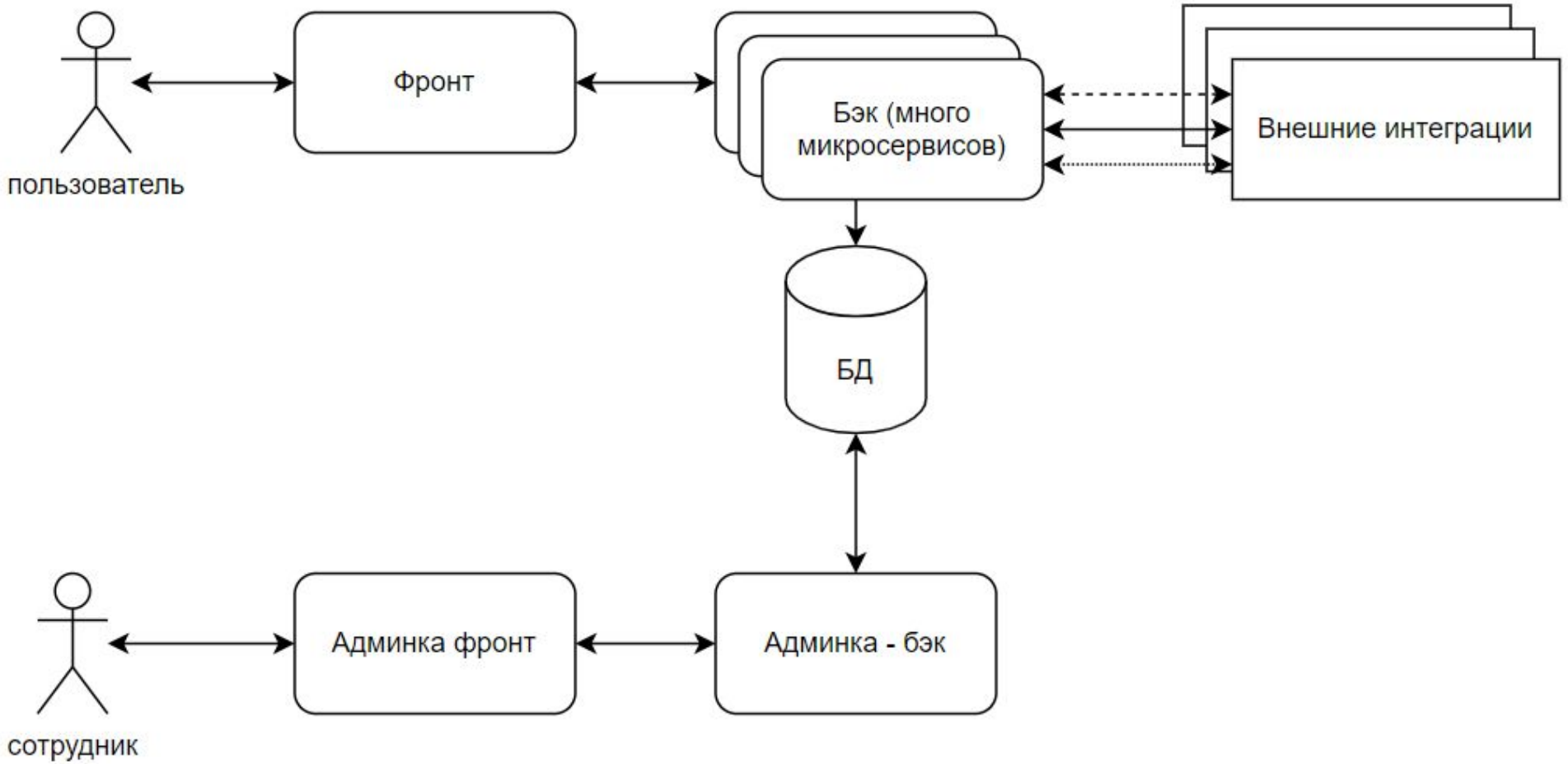
Модуль 1

Понятие системы

- Система (ИТ) - набор программно-аппаратных компонентов, предоставляющий определенные услуги. В более широком смысле также включает персонал, организационную структуру, процессы.
- Контекст — та часть внешней среды, понимание которой необходимо для работы с системой и определения требований к ней.
- Скоуп — зона ответственности(проекта), все то, что может дорабатываться вашей командой для достижения целей проекта.
- Сложность корпоративных ИТ систем — практически любой бизнес процесс включает десятки ИТ систем, каждая система — участвует в десятках процессов.
  - Если добавить к этому постоянный процесс доработки существующих систем становится понятно, что процесс анализа заканчивается только с выводом системы из эксплуатации.

Модуль 1

Типичная структура ИТ системы



Модуль 2

Документация: виды документации

2

ак. ч.

- По назначению
  - Для разработки
  - Для сдачи работ заказчику
  - Пользовательская документация
  - Для партнеров по интеграции
  - Для техподдержки и администраторов
- По типам требований
  - Функциональные (что система делает)
  - Нефункциональные (нагрузка, безопасность, совместимость)
  - Требования к миграции

Модуль 2

Документация: структура технического задания

- Реквизиты: версия, автор, назначение, проект, изменения от предыдущей версии, согласования
- Глоссарий
- User Story (кто, что, зачем) и/или бизнес требования и цели
- Интерфейс (фронт)
- Алгоритмы обработки данных (при необходимости модели и диаграммы)
- БД (таблицы, функции и процедуры) и маппинг данных
- API
- Обработка ошибок, исключений, справочник ошибок
- Справочники/внешние ссылки

Модуль 2

Структура задачи на разработку(тикета)

- Задача в таск трекере
  - Описывается отдельная, небольшая часть общего плана работ, которую можно отдельно протестировать
  - Указывается версия, срок внедрения, приоритет
  - От каких задач зависит и какие блокирует
- Важно делать отдельные задачи для:
  - Фронта
  - Бэка
  - Администраторов и др.
- К задаче прикладываются
  - тест кейсы и/или сценарии использования
  - Ссылка на документацию (например, в Конфлюенс)

Модуль 2 - User story и Use case

- **User Story**
- Краткое описание задачи «на салфетке» - кто делает что зачем?
- Не указываем подробности, только самое важное для бизнес цели
- Занимает максимум несколько абзацев
- Цель — понять, какую задачу мы решаем

- **Use case**
- Подробное пошаговое описание процесса с учетом всех возможных событий — основной поток когда все работает, альтернативный поток, когда возникла ошибка, необходимость произвести дополнительные действия и т. п.
- Занимает несколько страниц
- Цель — подготовить исчерпывающее описание для разработки, тестирования и приемки

Модуль 2

Маппинг данных

Обязательно нужна таблица с указанием, что мы сохраняем в БД и откуда это берем

Поле	Тип данных	Обязательность	Источник	Комментарий
id	int	+	Автогенерация	Уникальный идентификатор файла
name	varchar	+	Имя полученного файла	
received	dateTime	+	Время получения файла	
status	Enum (loaded/error)	+	Результат обработки файла	Статус может отличаться от результата обработки файла, если произошла ошибка записи в БД
error_code	varchar	-	Код ошибки	См справочник Коды ошибок.
error_desc	varchar	-	Описание ошибки	См справочник Коды ошибок.

Модуль 2

Тестирование

- Помним о тестировании — требования должны давать возможность готовить тест кейсы:
  - для всех принимаемых данных указывается тип данных, диапазон значений
  - указывается обязательность/необязательность наличия значений
  - указывается ожидаемый объем данных (например,на ближайшие несколько лет)
  - описываются все возможные ситуации, а не только наиболее вероятные

Модуль 2

Задание

Подготовить задачу в Jira для разработчика бэка:

- вход в приложение по логину и паролю (с учетом ограничений на кол-во попыток указать пароль) при условии, что пароль уже установлен пользователем
- Подготовьте описание алгоритма работы текстом или схемой (обратите внимание на момент сброса счетчика попыток входа)
- Дополните тест кейсами в формате действие — ожидаемый результат
- Укажите типы принимаемых данных в виде таблицы
- Что храниться в БД (подсказка - хеш)

Модуль 3

Интеграция автоматизированных систем

6

ак. ч.

- Что такое интеграция и зачем она нужна
- На какие параметры обращают внимание при проектировании интеграций
- Виды интеграций
  - Файловый обмен
  - Общая БД
  - WEB API (REST)
  - Очереди
  - Удаленный вызов процедуры (RPC)
  - Проприетарные протоколы, стандарты и т.п.
- Тест

Модуль 3

API

- API — application programming interface — это набор команд, с помощью которых можно взаимодействовать с данной программной
- Команда API включает **название команды, название и типы входных данных** и их допустимые значение, а также **типы возвращаемых данны**х

Пример функции:

Деление(int делимое, int делитель) возвращает int *частное*, int *остаток*

Деление(19, 9): частное 2, остаток 1

Также имеет значение адрес — куда направить функцию, к какому компьютеру, на какой порт(адрес программы на компьютере)

Что такое интеграция и зачем она нужна

- Системная интеграция — это способ организации информационного обмена между АС, набор команд, которые одна АС выполняет по запросу другой:
  - Формат данных, переменные и методы
  - Канал передачи данных
  - Последовательность запросов и ответов
  - Обработка ошибок
- Получает распространение подход открытых API
- Получает распространение микросервисный подход к архитектуре
- Даже в небольшой компании в рамках одного БП информация проходит через несколько АС, а в крупных — через десятки.

На какие параметры обращают внимание при проектировании интеграций

- Насколько мы доверяем другой АС? Другая АС принадлежит нашей организации, партнеру или открыта всему интернету?
- Можно ли будет менять процесс/API в одностороннем порядке, как его дорабатывать, исправлять ошибки
- Как обеспечивается безопасность
- Простота разработки, поддержки
- Нагрузка
- Формат данных (текстовый/бинарный)
- Формат взаимодействия — запрос - ответ/по расписание/поддержание состояния
- Обработка нештатных ситуаций

Файловый обмен

- Часто применяется когда нужно передать информацию по расписанию
- Форматы данных: csv, xml, txt
- Используется общая директория и протокол доступа к ней (SFTP, smb)
- Иногда используется почта
- Часто для подтверждения подлинности и юридической значимости информации используется электронная подпись
- На практике важно предусмотреть обработку множества ситуаций:
  - Файл не получен
  - Получено несколько файлов(с одним именем)
  - Формат файл некорректен
  - Не удалось прочитать файл

Общая БД

- Может использоваться только в рамка одной организации:
  - Очень проста в реализации
  - Сильная взаимосвязь двух систем
  - Одна система может блокировать или нагружать таблицы для другой
- Подразумевает сильную зависимость двух АС, чего следует избегать
- Использование транзакций, пакетной обработки, статусов (иногда получаем аналог очереди)
- Часто используется для отчетов и аналитики

WEB API (REST)

- Максимальное использование HTTP/HTTPS
- Обращение — запрос - ответ (**синхронное**)
- Формат передаваемых данных — текстовый (чаще всего JSON/XML)
- Преимущества:
  - Простота для разработчика
  - Фактический стандарт с точки зрения популярности
  - Отделяет сервер от клиента
  - Кеширование
- Недостатки:
  - Не поддерживает состояние
  - Не совсем стандарт, технически многое нужно предусмотреть самостоятельно, не очевидно как соотнести HTTP с бизнес задачами

Очереди

- Включают отправителя, среду доставки, маршрут, получателя. Главное — отделяет отправителя от получателя.
- Основные типы взаимодействия:
  - Публикация - подписка
  - Запрос-ответ
- Широко распространены в рамках одной организации
- Обеспечивают хорошую изоляцию АС друг от друга
- Много различных вариантов работы и форматов данных (xml, protobuf) в зависимости от назначения
- Подходят когда кол-во получателей\отправителей не определено или нужна сложная **маршрутизация сообщений по правилам**
- Реализуются специальным ПО (IBM MQ, Kafka, Rabbit MQ)

Удаленный вызов процедур (RPC)

- Одна программа вызывает метод (процедуру), который должен быть выполнен в другой программе, возможно, на другом сервере
- Существует множество реализаций, можно выделить:
  - Различные реализации в рамках конкретных языков программирования (например Java RMI)
  - SOAP — протокол на базе HTTP/очередей с использованием XML/XSD для описания методов, сложный в описании, медленный, но за счет формализации меньше вероятность ошибки при использовании
  - XML-RPC, JSON-RPC, gRPC — сравнительно простые протоколы, которые, в частности, позволяют реализовывать двустороннее взаимодействие — т. е. сервер может сам отправлять данные клиенту без запроса, а по событию. Это полезна для игр, биржевых приложений, оповещений

Краткие выводы

Тип интеграции	Для внешней АС	Формат данных	Тип взаимодействия
Файловый обмен	да	xml, csv,txt	запрос/ответ по расписанию
Общая БД	нет	строки в таблице	заполнение таблицы, чтение из нее
Очередь	нет	зависит от типа очереди	запрос-ответ/публикация - подписка
WEB API	да	json/xml	запрос/ответ
gPRC, JSON-RPC, XML-RPC	да	xml/json	запрос/ответ или двусторонний обмен сообщениями

Какой тип интеграции вы используете в следующих случаях

1) Социальная сеть позволяет разработчику стороннего приложения узнать список друзей определенного пользователя

2) Раз в день банк выгружает список проведенных транзакций контрагенту для сверки

3) Внутри банка все совершенные сделки передаются во внутреннюю систему формирования банковской выписки

4) Заказы пользователя передаются от сервиса обработки заказов на сервис отправки письма с подтверждением заказа

5) Поставщик передает в магазин товары для добавления в каталог

6) Пользователь получает уведомление о новом сообщении в мессенджере

REST API — HTTP

- HTTP включает
  - Стартовая строка (метод URI HTTP/Версия), например: GET /test/users
  - Заголовки, например content-type: text/html; charset=windows-1251
  - Тело сообщения (не обязательное, отсутствовует у метода GET)
- Методы:
  - GET — получение ресурса
  - POST — создание ресурса
  - PUT — обновление ресурса
  - DELETE — удаление ресурса (другие методы есть, но практически никогда не используются)
- Коды ответов:
  - 1xx – информация, 2xx – успех, 3xx- редирект, 4xx – ошибка клиента, 5xx – ошибка сервера
- Явно указываем на сервере, можно ли клиенту кешировать ответ

REST API — HTTP

- GET – получить данные данные
  - Можно передавать параметры запроса в uri
  - Пример
    - GET http://www.test.com/api/v1.0/users
    - GET http://www.test.com/api/v1.0/users/?deleted=true&limit=100
- POST – создать объект
  - Пример POST http://www.test.com/api/v1.0/users
- PUT – создаёт новый ресурс или перезаписывает существующий
- Важные характеристики запроса:
  - Идемпотентность – повторное направление одинакового запроса не имеет последствий (GET, HEAD, PUT, DELETE, HEAD, OPTIONS)
  - Безопасность – метод ничего не меняет на сервере, а только читает данные(GET, HEAD,OPTIONS)

REST API - GET и POST

- GET
- Body не заполнено
- Параметры запроса — в строке запроса - GET http://www.test.com/api/v1.0/users/?deleted=true&limit=100
- Длинна строки запроса ограничена
- Строка запроса может кешироваться, сохраняться в логах (а значит в нельзя передавать пароли и т.п)

- POST
- Параметры запроса передаем в Body
- Body не кешируется
- Можно передать много данных
- Часто используем по умолчанию, если GET не подходит

REST API— что включаем в описание

- HTTP методы (GET, POST..)
- Описание параметров запроса/ответа (таблица с типами данных и/или XML/JSON схема)
- Пример запроса и ответа
- url (включая адреса хостов для теста и прода)
- Коды ответов (200 OK, 403 Forbidden и др.) и описание ошибок
- Нагрузка (ожидаемая и максимально допустимая)
- Аутентификация (пароль, токен, сертификат)

REST API— XML

- XML - это формат описания данных
- Хорошо подходит для
  - представления иерархических данных
  - контроля за форматом получаемых данных
- Нужно помнить о:
  - кодировке (encoding="UTF-8" по умолчанию, но не всегда)
  - служебных символах, которые нужно заменять или не использовать:

символ	замена
<	<
>	>
&	&
' (в атрибуте)	'
" (в атрибуте)	"

REST API - XML

- Структура XML документа:
  - Пролог — тип документа: xml, версия xml, кодировка
  - Корень — обязательно строго один
    - Иерархически организованные элементы внутри корня
- Пример:

```
<?xml version="1.0" encoding="utf-8"?>
<person>
    <Фамилия>Иванов</Фамилия>
    <Имя>Иван</Имя>
    <Год_рождения>1980</Год_рождения>
</person>
```

REST API— JSON

- JSON - это формат представления данных, функциональный аналог XML
- Хорошо подходит для
  - представления иерархических данных
  - контроля за форматом получаемых данных
- Проще, чем XML, но имеет меньше возможностей для контроля данных
- Основные элементы:
  - множество пар ключ:значение, например {"name":"John", "age": 31}
  - Массив, например ["Moscow","London","Paris","Berlin"]
  - Число, например 45 или 0.8
  - Булевое значение true или false
  - Строка, например "Test"
  - null — пустое значение

SOAP — REST сравнение подходов к API

Параметр	SOAP	REST
Базовая идея	RPC - вызвать функцию указав ее имя и параметры	REST - обратиться к объекту по определенному url как если бы это была веб страница
Транспортный протокол	HTTP, очередь или другой, не важно	Только HTTP
Структура передаваемых данных	Только XML	XML, JSON могут быть и другие варианты
Метод HTTP	POST без вариантов (если используем HTTP)	GET/POST/PUT/DELETE или любой другой HTTP метод
Использование схемы	Обязательно	Опционально
Схема	Стандартизированная протоколом	Любая, нет стандарта (есть OpenAPI но это не едиственный вариант

REST API— задание

Подготовить описание REST API, с помощью которого пользователи цента Специалист смогут указывать свои паспортные данные и данные об образовании. Сохраненные данные также можно запросить. (Это позволит автоматически заполнять паспортные данные в договорах, а документы о высшем образовании использовать для оформления повышения квалификации.)

- Пользователь API — фронт сайта, который получит информацию от пользователей и сохранит ее на бекенде, а после сохранения — покажет пользователю в его профиле.
- Цель описания — показать разработчику фронта сайта — как добавить данные, чтобы бекенд их сохранил, и как получить их обратно от бекенда.
- Предусмотреть поведение, если запрос некорректен по формату
- Можно выбрать xml или json для передачи данных

Модуль 3

Надежность и нагрузка — основные понятия

- RTO (recovery time objective) – это максимальное допустимое время восстановления данных — т. е. сколько времени простоя системы допустимо. Увеличивается за счет «холодного» и «горячего» резерва и дублирования.
- RPO (recovery point objective) - это максимальный период времени, за который могут быть потеряны данные в результате инцидента. Увеличивается за счет резервного копирования.
- SLA (Service Level Agreement) - договор между заказчиком услуги и её поставщиком, содержащий:
  - Время доступности системы (например 95% времени в рабочие дни)
  - Допустимое среднее и максимальное кол-во запросов в секунду, час, день и т. п.
  - Время отклика и т.п.

Модуль 3

Безопасность

- Помнить о персональных данных (какие данные, в каком объеме будут в нашей системе) — в зависимости от этого могут потребоваться дополнительные меры защиты и архитектурные ограничения.
  - Закон "О персональных данных" от 27.07.2006 N 152-ФЗ
  - Постановление Правительства РФ от 1 ноября 2012 г. N 1119 "Об утверждении требований к защите персональных данных при их обработке в информационных системах персональных данных"
- Продумать процедуру регистрации и аутентификации пользователей как компромисс между безопасностью и удобством
- В целом помнить, что требования нужно согласовывать со специалистом по ИБ

Модуль 3

Основные методы аутентификации

- Логин и пароль отправляются при каждом запросе
- На основе логина и пароля клиент получает токен. Клиент отправляет этот токен с каждым запросом. Токен переодически менятся (запрашивается новый, истарый истекает)
- Сертификат HTTPS (TLS) — клиент сохраняет в настройках сертификат с данными сервера в качестве «доверенного». При установке соединение сервер подтверждает свою подлинность. Потверждение подлинности сервера основывается на проверке клиентом данных от сервера с использованием сохраненного сертификата.

Модуль 3

Ошибки и проблемы

Как системный аналитик вы должны предусмотреть в ТЗ, что делать, когда происходят следующие проблемы:

- одна из систем неработоспособна или нет сети
- система отвечает слишком долго
- ответ некорректен по формату
- поток сообщений слишком велик
- внешняя система не проходит проверку безопасности
- произошло что-то совсем непонятное(неизвестная ошибка)

Модуль 4

Моделирование

2

ак. ч.

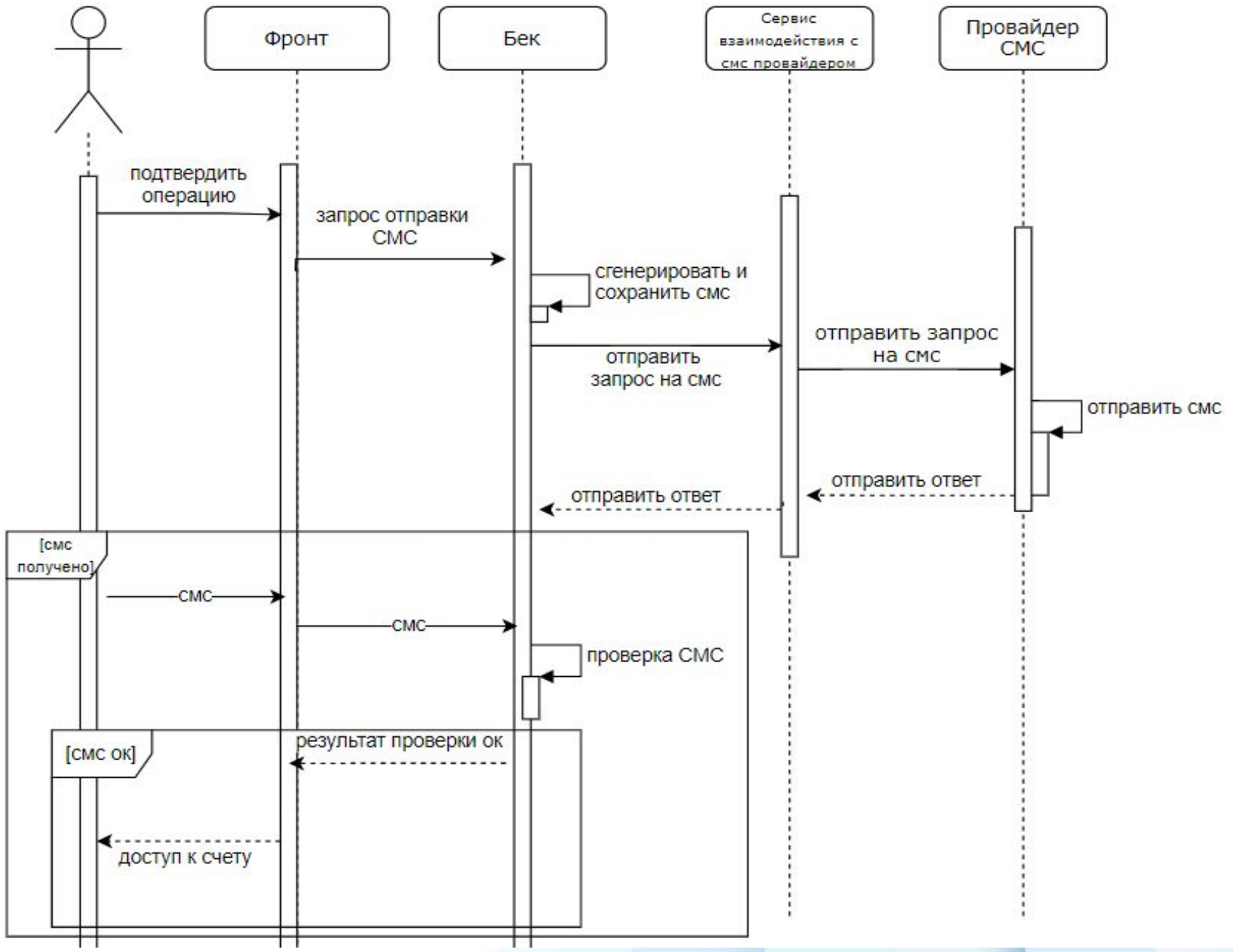
- Зачем нужно моделирование
  - визуально проще воспринимать сложные процессы и объекты в виде модели
  - При моделировании остаются только самые важные детали
- Распространенные нотации моделирования: UML, BPMN, IDEF0
- Распространенные виды моделей/структурированного представления информации
  - UML диаграмма последовательностей
  - UML диаграмма деятельности
  - ER-диаграмма/UML диаграмма классов
  - UML диаграммы состояний

Модуль 4

Моделирование - UML диаграмма последовательностей

Важно понимать разбиение системы на (микро)сервисы

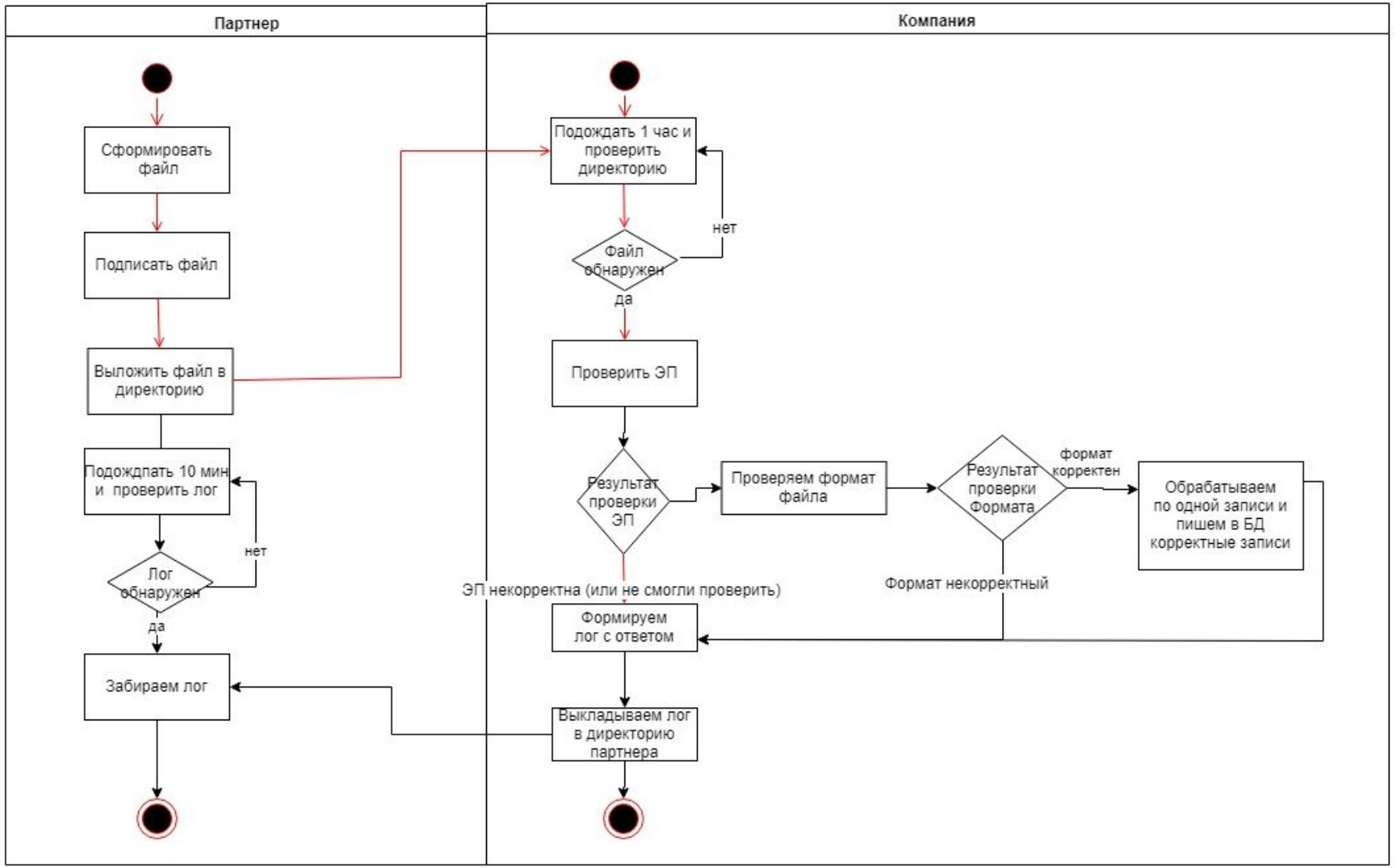
Пример — отправка СМС для входа в ЛК банковского приложения



Модуль 4

Моделирование - UML диаграмма деятельности

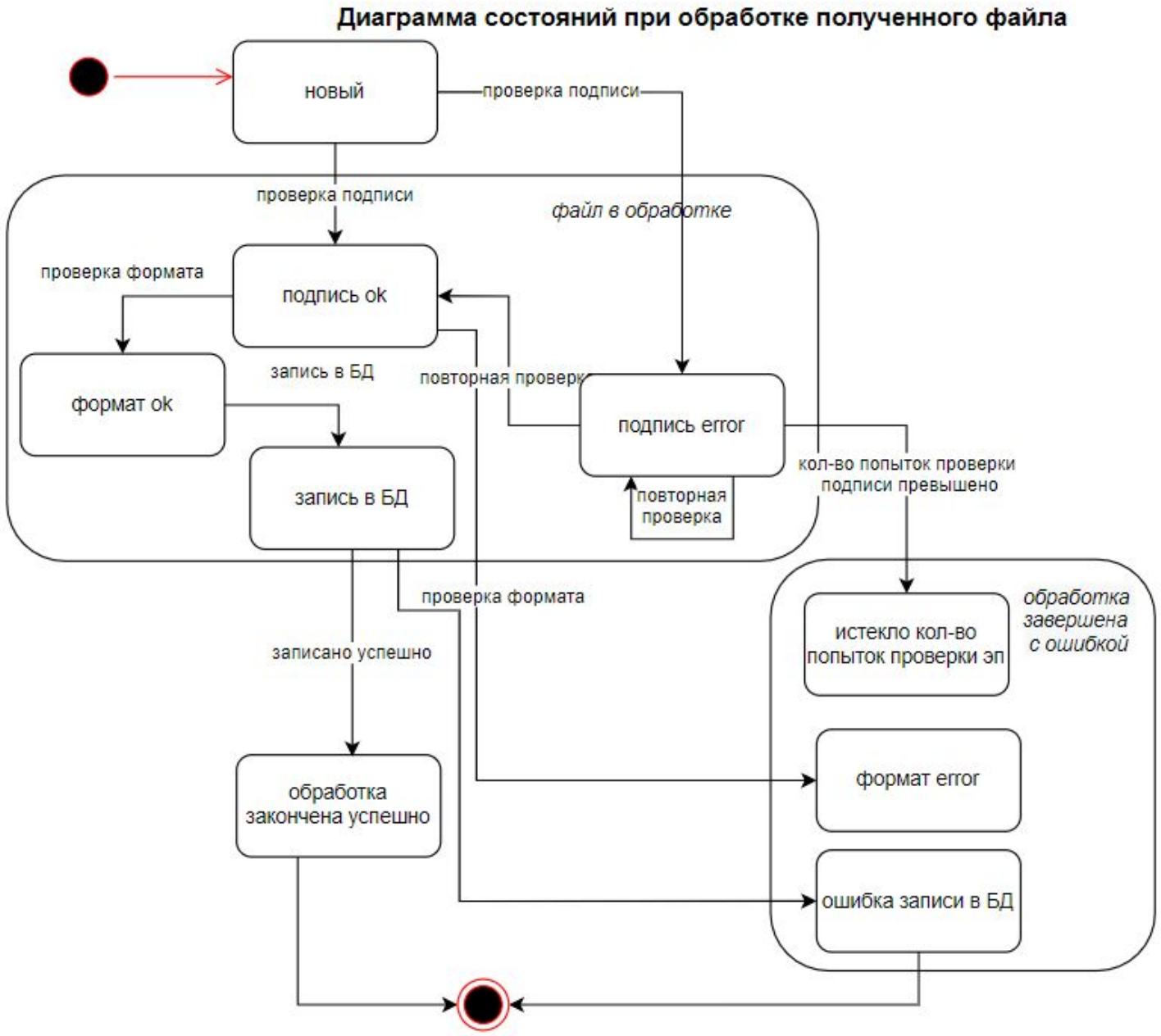
Пример — отправка файла между двумя компаниями



Модуль 4

Моделирование - UML диаграмма состояний

Пример — диаграмма состояний файла

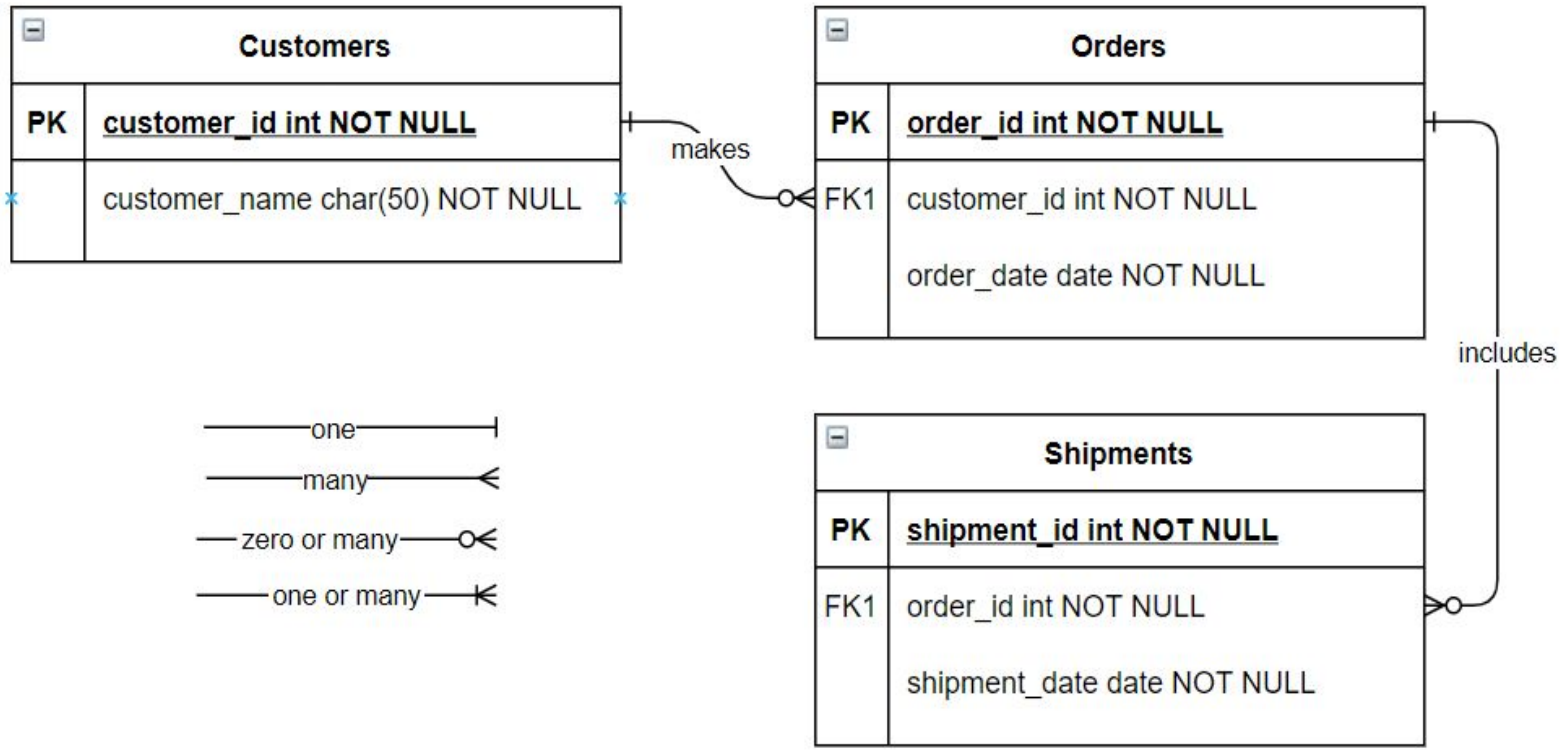


Модуль 4

Моделирование - ER-диаграмма — моделирование БД

Основные элементы:

- Сущности
- Связи между сущностями
- Кратность связей
- (один к одному,
- один ко многим, многие к



Модуль 4

Моделирование - задание

- Какие модели вы будете использовать для описания
  - Схемы заказа авиабилета через агрегатор
  - Работы будильника
  - Отправки посылки по почте
  - Хранения данных о заказанных книгах в БД

Модуль 4

Коммуникации

- Выявление стейкхолдеров
  - Список ролей в системе
  - Список пользователей и их должности
  - Организационная структура
  - Опрос сотрудников
  - Изучение функционала
  - Изучение инструкций
  - Изучение участников с переписке, Jira и т. п.

Модуль 4

Коммуникации

- Стратегия выстраивания отношений со стейкхолдерами:
  - экономить время людей
  - критически воспринимать полученную информацию
  - искать пути проверить информацию
  - при возникновении ошибок, недопонимания искать решение, а не виноватых

Модуль 4

Коммуникации

- Способы получения данных от стейкхолдеров:

- Общее обсуждение в группе
- Интервью один на один
- Опрос с перечнем вопросов
- Наблюдение за работой стейкхолдеров
- Фокус группа (менее распространена)
- Воркшоп (менее распространен)
- Эксперименты/АВ тестирование (скорее маркетинг, анализ данных)
- Согласование полученной информации — формальное и неформальное
- Согласование результата разработки - UAT (User Acceptance Testing)

Модуль 4

Источники информации:

- документация (но не надо сразу верить тому, что в ней написано)
- тестовая система (может не соответствовать прод версии)
- чтение кода и уточнение непонятных вопросов и у разработчика
- стейкхолдеры (самый важный источник, но не всегда доступный)
- трекеры задач — Jira, Redmine, HPSM и .т.п.
- законодательство и подзаконные акты
- наблюдение за пользователями
- можно попросить администраторов прислать листинг конфигураций, директорий, список хостов

Модуль 4

Обработка информации:

Этап 1

- составить описание всех входящий и исходящих потоков
- с какими структурными подразделениями взаимодействует система
- какие у системы роли и сколько пользователей
- уровень требований к ИБ
- конфигурация с точки зрения железа и ПО
- типичные проблемы (причины инцидентов)
- на этом этапе возможно следует уточнить задачу, зону ответственности

Модуль 4

Обработка информации:

Этап 2

Поиск противоречий в полученных данных:

- всегда задаем себе и другим вопрос — **зачем**.
- для принципиально важной информации — **не менее двух независимых источников**
- полученные схемы, описания следует показать стейкходерам, попытаться получить их **подтверждение**
- используйте научный метод — если функционирование/назначение элемента непонятно — **выдвигаем гипотезу и тестируем ее**
- состояние как есть не обязательно корректно
- ранее полученные результаты быстро устаревают.

Модуль 5

Практика - задание

3
ак. ч.

Подготовка ТЗ по функционалу:

- Добавить на сайт специалист возможность пользователям указывать свои паспортные данные и данные об образовании, чтобы (в другой доработке) автоматически заполнять паспортные данные в договорах, а документы о высшем образовании использовать для оформления повышения квалификации
- Описание API — уже сделано в предыдущем задании, теперь надо оформить логику того, как бэк проверяет и сохраняет данные. Кроме того, нужно оформить ТЗ в целом, написать про назначение системы, user story, чтобы бэк мог это реализовать, а тестировщик — протестировать.
- Возникающие доп вопросы — задавайте преподавателю или можете сами придумать, как на них ответили бы стейкхолдеры

Список рекомендованной литературы

- Проектирование веб-API. Арно Лоре
- Шаблоны интеграции корпоративных приложений. Грегор Хоп, Бобби Вульф
- Разработка требований к программному обеспечению. Карл Вигерс, Джой Битти
- Применение UML 2.0 и шаблонов проектирования. Крэг Ларман
- Высоконагруженные приложения. Клеппман Мартин

Документы об окончании

- Свидетельство Центра «Специалист» об окончании обучения

- Свидетельство Центра «Специалист» международного образца

- Удостоверение о повышении квалификации

Спасибо за внимание

Ваши вопросы

Учебный центр «СПЕЦИАЛИСТ» – Ваш путь к успеху

info@specialist.ru

+7 (495) 232-32-16